Numer Algor
DOI 10.1007/s11075-017-0274-4

ORIGINAL PAPER

Quasi-analytical root-finding for non-polynomial functions

Cleonice F. Bracciali[1] · **Michael Carley**[2]

Received: 17 May 2016 / Accepted: 18 January 2017

Abstract A method is presented for the calculation of roots of non-polynomial functions, motivated by the requirement to generate quadrature rules based on non-polynomial orthogonal functions. The approach uses a combination of local Taylor expansions and Sturm's theorem for roots of a polynomial which together give a means of efficiently generating estimates of zeros which can be polished using Newton's method. The technique is tested on a number of realistic problems including some chosen to be highly oscillatory and to have large variations in amplitude, both of which features pose particular challenges to root–finding methods.

Keywords Root-finding · Orthogonal functions · Quadrature rules

1 Introduction

Root–finding is one of the classical problems in numerical analysis, and continues to stimulate research. Our motivation in this paper is to develop methods for the roots of functions with distinct real roots. Our particular application comes from the study of orthogonal functions, both classical and of types with more general orthogonality properties [3, for example], though we emphasize that our method is not limited to

✉ Michael Carley
m.j.carley@bath.ac.uk

Cleonice F. Bracciali
cleonice@ibilce.unesp.br

[1] Departamento de Matemática Aplicada, UNESP–University Estadual Paulista, 15054-000 São José do Rio Preto, SP, Brazil

[2] Department of Mechanical Engineering, University of Bath, Bath BA2 7AY, UK

Published online: 21 February 2017

these functions and applies to any function which can be represented locally by a Taylor series.

A concrete motive for finding the roots of orthogonal functions is the derivation of quadrature rules, including quadrature rules on the unit circle. The classical orthogonal polynomials—Legendre, Jacobi, Hermite—have well-known quadratures associated with them and a correspondingly extensive body of work on efficient methods for their roots. There are also quadrature rules based on the orthogonality properties of special functions, such as the Bessel-function rules used for the evaluation of Hankel transforms.

While various methods exist which exploit particular special properties of the classical orthogonal functions, the methods of choice for general functions are probably those based on the use of ordinary differential equations for the function. The original form of this algorithm is that of Glaser, Liu, and Rokhlin [8], whose technique uses the second-order ordinary differential equation for the function to generate high-order approximations for stepping between successive roots. Their method uses a Prüfer transform to step from one root to an approximation of the succeeding root, which is then polished using Newton's method applied to a high (thirty or more) order Taylor expansion of the function. The coefficients of the Taylor expansion are efficiently generated using a recursion derived from the differential equation, which allows the function to be evaluated from its Taylor expansion to near machine precision.

The o.d.e. techniques have since been extended using the theory of non-oscillatory phase functions [6] with a recent paper by Bremer [5] presenting a root-finding algorithm which yields roots of the function in $O(1)$ operations per root after a set-up operation which requires very little computation. The algorithm also allows parallel computation of roots, unlike the previous version [8] which required that roots be found sequentially.

Another approach to root-finding for general functions [2] is to represent the functions as combinations of Chebyshev polynomials on sub-intervals of the appropriate domain. Roots, if any, can then be found on each sub-interval as the eigenvalues of the companion matrix of the Chebyshev approximation. This gives good approximations to the roots but does require that the interpolation be generated in advance.

While o.d.e. techniques are probably most convenient for many functions, they do require that a differential equation be available for the stepping between roots, and for generation of the Taylor expansion. When the differential equation is not available, we are forced back on more classical root-finding methods with the corresponding difficulties of root isolation and accurate evaluation. In this paper, we make use of the Taylor series to step from root to root, and as a means of isolating approximations to roots which can then be polished using Newton's method. The approach is not as powerful as the o.d.e.-based algorithm, but is more flexible in that it can be used when a differential equation is either not available or is computationally difficult to use for expansions.

Our root-finding method is based on a combination of two ideas. The first is the use of truncated Taylor series to approximate a function near some point. This is valid over a finite interval whose size depends on the order of terms retained in the Taylor series. Over that interval, the expansion is an accurate representation of the underlying function, to within some user-specified tolerance. The second idea is the

use of Sturm sequences to find a root of the Taylor series in the interval. Our argument is that if the Taylor polynomial faithfully represents the function over an interval, then its roots in that interval are good approximations to the roots of the function proper.

2 Algorithm

The underlying principle of the root-finding algorithm is to locally replace the function $f(x)$ whose distinct real roots are to be found with a polynomial approximation of controlled accuracy. In principle, if the Taylor polynomial (truncated Taylor series) is accurate enough over some range, roots of the Taylor polynomial will be close approximations of roots of the underlying function $f(x)$. Our method is thus composed of two basic tools: the Taylor polynomial as a local representation of the function, and the Sturm sequence which can be used to isolate roots of the Taylor polynomial as initial guesses for the roots of the function proper.

2.1 Taylor polynomials

The Taylor polynomial $T_N(\Delta x, x_0)$ of order N is an approximation of a function $f(x)$ in the vicinity of a point x_0,

$$\begin{aligned} T_N(\Delta x, x_0) &= \sum_{n=0}^{N} t_n (\Delta x)^n, \\ t_n &= \frac{1}{n!} \left. \frac{\mathrm{d}^n f}{\mathrm{d}x^n} \right|_{x=x_0}, \\ \Delta x &= x - x_0 \end{aligned} \tag{1}$$

with a corresponding estimate of the remainder,

$$\begin{aligned} R_N(\Delta x, x_0) &\approx \frac{1}{(N+1)!} \left. \frac{\mathrm{d}^{N+1} f}{\mathrm{d}x^{N+1}} \right|_{x=x_0} (\Delta x)^{N+1}, \\ |f(x) - T_N(\Delta x, x_0)| &< R_N(\Delta x, x_0). \end{aligned} \tag{2}$$

We will use this error estimate to fix the range of validity of the Taylor polynomial used in approximating the function whose roots are to be found.

Our method does not depend on any particular means of generating derivatives for the coefficients of the Taylor polynomial, and to some degree a choice of method will depend on the application. In the approach closest to ours [5, 8], an o.d.e. is used to generate the required information about the function: our aim here is to provide a method which can be used when an o.d.e. is not readily available. We note that in the general case, high-order derivatives of an analytic function can be stably and accurately found using numerical evaluation of Cauchy integrals [1] though in practice we believe that in most applications there is usually enough local information about the function to provide efficient methods for the computation of derivatives without recourse to Cauchy integrals.

In the applications which motivate this work, the function $f(x)$ is often defined by a sequence of functions $\{f_n(x)\}$ which satisfies a three–term recursion of the form

$$f_{n+1}(x) = a_n(x) f_n(x) + b_{n-1}(x) f_{n-1}(x), \tag{3}$$

so that one means of generating the derivatives required for the Taylor polynomial is repeated differentiation of the recursion,

$$\frac{\mathrm{d}^m}{\mathrm{d}x^m} f_{n+1}(x) = \sum_{q=0}^{m} \binom{m}{q} \left(\frac{\mathrm{d}^q}{\mathrm{d}x^q} a_n(x) \frac{\mathrm{d}^{m-q}}{\mathrm{d}x^{m-q}} f_n(x) + \frac{\mathrm{d}^q}{\mathrm{d}x^q} b_{n-1}(x) \frac{\mathrm{d}^{m-q}}{\mathrm{d}x^{m-q}} f_{n-1}(x) \right). \tag{4}$$

Although this is the form of recursion used for orthogonal polynomials, for which b_{n-1} is independent of x, there is no requirement that a_n or b_{n-1} be polynomial, and we impose no such restriction. For example, the Bessel function recursion makes use of rational functions [9, 8.471],

$$\mathscr{Z}_{\nu+1}(x) = \frac{2\nu}{x} \mathscr{Z}_{\nu}(x) - \mathscr{Z}_{\nu-1}(x),$$

and in our numerical tests we will include functions defined by recursions based on algebraic functions.

2.2 Sturm sequences

The Sturm sequence is a method for the isolation of roots of a polynomial on the real line, dating to the nineteenth century [13] and which has previously been implemented for use in a polynomial root-finding algorithm based on bisection [10]. Sturm's theorem states that for any polynomial $p(x)$, with simple real roots, there exists a sequence of polynomials $p_i(x)$, $i = 1, 2, \ldots, m$. At any point x, the function $s(x)$ can be calculated as the number of sign changes in the sequence $p_i(x)$. The number of roots lying between $x = a$ and $x = b$, with $a < b$, is then the difference $s(a) - s(b)$. This gives us a means of determining the number of roots of our Taylor polynomial as a first step towards finding roots of $f(x)$.

The Sturm sequence for any polynomial $p(x)$ with simple real roots is generated by initializing the sequence with $p_1(x) = p(x)$, $p_2(x) = p'(x)$. Subsequent terms $p_k(x)$ are generated as the remainder of the polynomial division $p_{k-1}(x)/p_k(x)$ with its sign reversed. An implementation of this algorithm in C has been published [10] and consists of the following steps for a polynomial $p(x) = \sum_{n=0}^{N} a_n x^n$:

1. Set $p_1(x) = p(x)$ and $p_2 = p'(x)$;
2. Scale $p_2(x)$ on the absolute value of its leading coefficient;
3. Set $k = 3$.
4. While order$(p_k(x)) \neq 0$:
   (a) Set $p_k(x) = -$remainder $(p_{k-2}(x)/p_{k-1}(x))$;
   (b) Scale $p_k(x)$ on the absolute value of its leading coefficient;
   (c) Set $k = k + 1$.
5. Set $m = k$.

The output of this algorithm is a sequence of m polynomials which can be evaluated at a point x to check for sign changes as follows:

1. Set $s(x) = 0$;
2. For $k = 2, 3, \ldots, m$:
   (a) $a = p_{k-1}(x)$, $b = p_k(x)$;
   (b) if $ab < 0$ or $a = 0$, increment $s(x)$.

Given these algorithms for evaluation of the Sturm sequence and for counting sign-changes, we can check for and, if necessary, bracket one root in an interval $0 < x < h$, by the following bisection method, where α is a desired interval size for the bracketing:

1. Apply the Sturm sequence at $x = 0$ and $x = h$, if $s(0) = s(h)$ terminate and report no root;
2. Set $x_{\min} = 0$, $x_{\max} = h$, $s_{\min} = s(0)$, $s_{\max} = s(h)$;
3. Set $x_1 = (x_{\min} + x_{\max})/2$, $s_1 = s(x_1)$;
4. If $s_1 \neq s_{\min}$, set $x_{\max} = x_1$, $s_{\max} = s_1$;
5. Otherwise, set $x_{\min} = x_1$, $s_{\min} = s_1$;
6. If $|x_{\max} - x_{\min}| < \alpha$ and $|s_{\max} - s_{\min}| = 1$, terminate;
7. Return to step 3.

The algorithm isolates the root in $[0, h]$ with smallest absolute value, i.e., the 'next' root in our stepping procedure. Replacing h with $-h$ in the first two steps gives a procedure which steps in the negative direction, which allows our root-finding algorithm to be applied starting from an arbitrary point. For example, this allows the method to work forwards and backwards from some starting point to find all roots of a function in a given interval, even when there is no symmetry or other information available about the distribution of roots. This ability to start from an arbitrary point means that the method can also be implemented in parallel to seek multiple roots independently.

2.3 Summary

To summarize, we give an algorithm for stepping from an initial point x_0 to the next largest root of $f(x)$, $f(x_0 + \Delta x) = 0$. This step can be repeated to find subsequent roots, and reversed to find roots $x_0 - \Delta x$. The user supplies a starting point, which may be a previously-found root, and an error tolerance ε which is used to control the precision of the Taylor expansion by setting the range over which a root is sought.

To step from point x_0 to the next largest root, using expansions of order N:

1. Localize region containing roots:
   (a) Generate $T_N(\Delta x, x_0)$;
   (b) Generate Sturm sequence for $T_N(\Delta x, x_0)$, using method of Section 2.2;
   (c) Evaluate step size $h = (\varepsilon/|t_N|)^{1/N}$;
   (d) Use Sturm sequence to check for roots in interval $0 \leq \Delta x \leq h$;
   (e) If no roots, set $x_0 = x_0 + h$ and repeat, otherwise terminate.

2. Isolate one root:

   (a) When interval containing roots is found, use bisection method of Section 2.2 to isolate one root Δx;
   (b) Refine root using Newton's method;
   (c) Set x_0 to $x_0 + \Delta x + \delta$ and repeat.

The tolerance ε is user-defined and will depend to some degree on the problem being solved. The small increment δ is required to shift the starting point off the root which has just been found in order to take that root out of the interval covered by the Sturm sequence, to machine precision.

2.4 Some notes on implementation

In our numerical examples, we consider functions which have amplitudes comparable to machine precision, a stringent test of a numerical root-finding method. In our experience, the choice of N, ε, and δ affect the performance of the algorithm and we outline how they can be set.

Firstly, the order N of the method interacts with the tolerance ε in setting the step size h. At high order, small tolerance, and large $|t_N|$, h cannot be reliably evaluated, limiting the maximum usable order. We find that priority should be given to ε and the order adjusted rather than *vice versa*. Given that

$$h^N = \varepsilon/|t_N|$$

a usable criterion is to fix a minimum value of h^N, one or two orders of magnitude greater than machine precision say, and use this to fix a minimum value for $\varepsilon/|t_N|$ which will allow accurate evaluation of h. In our second test case, ε varies over many orders of magnitude at various values of N and the method still gives accurate results.

Secondly, too small a value of δ risks making the method fail on functions of very small amplitude. In our implementation, we shift x_0 by repeatedly incrementing it by δ until $|f(x_0)| > f_{\min}$, with $f_{\min}$ some small tolerance set by the user. Values of these settings are given in the principal tests which we present.

Finally, in principle, there might be functions for which the method of Section 2.3 fails to terminate. We have not encountered this problem in our tests, but for completeness the algorithm should stop and report failure if the number of steps exceeds some user-defined limit.

3 Numerical examples

We present a number of numerical examples intended to illustrate the behaviour of our algorithm and to show its performance on the kind of problem for which it is intended. The algorithm of Section 2.3 is applied with the Newton refinement step used to evaluate roots to near machine precision in all cases.

3.1 Trigonometric plus Gaussian function

Our first test case is used to graphically illustrate the behaviour of the method on a simple function which has closely-spaced roots or near-zeros which are not roots. The function is a combination of a Gaussian and a trigonometric function,

$$f(x) = \cos x - (1 + a)\mathrm{e}^{-x^2/\sigma^2}, \tag{5}$$

which can have a pair of zeros near $x = 0$ as well as the usual zeros of $\cos x$. The parameter σ controls the spacing of these zeros while the value of a determines whether or not they exist. For $a < 0$, there are no zeros near the origin, though the function may come very close to zero, while for $a > 0$ there are two zeros which can be made arbitrarily close. Figure 1 shows the behaviour of the function for different values of σ when there are two roots near the origin.

Figure 2 shows the behaviour of the algorithm on the test function for three different values of σ when there are two well-separated roots near zero ($a = 1/4$) and when the function approaches, but does not touch, the axis ($a = -1/128$). In each case, the root-finding was initialized at $x = -2$. Four roots were sought for $a = 1/4$, and two for $a = -1/128$, using eighth-order Taylor polynomials with $\varepsilon = 10^{-10}$ and $\delta = 10^{-12}$. The roots are indicated by solid boxes, while intermediate values of the function at the points x_0 used to generate Taylor expansions are shown as circles. The first obvious point to note is that the method does indeed locate the roots accurately. The second point of interest is that the adaptivity is clearly shown by the varying density of the circles on the curve, especially near maxima and minima. This is especially pronounced for $\sigma = 1/4$, where the circles are quite dense near the peaks of the function, and separate as the curve descends through the two central roots before bunching again around the minimum.

3.2 Highly-oscillatory trigonometric function

The next test case is $f(x) = \sin(1/x)$ which is chosen for being highly oscillatory making root-finding a challenging task as $x \to 0$. Roots of the function for $x > 0$

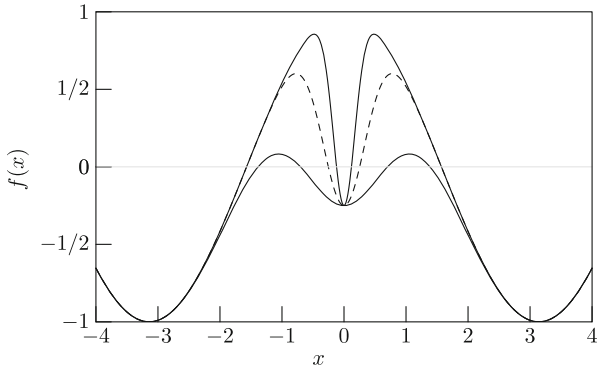

Fig. 1 Test function of (5), $a = 1/4$, $\sigma = 1, 1/2, 1/4$

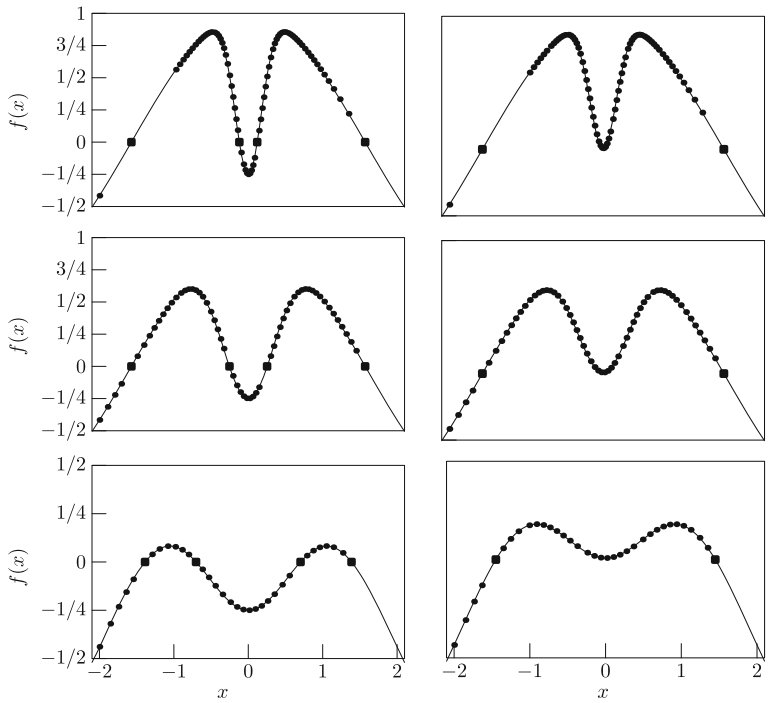

Fig. 2 Root-finding for trigonometric–Gaussian test function, *left hand column*, $a = 1/4$; *right-hand column*, $a = -1/128$; *top to bottom*, $\sigma = 1/4, 1/2, 1$; *squares* indicate roots; *circles* indicate expansion points

are $x_n = 1/(n\pi)$, $n \in \mathbb{N}$, so that the spacing between successive roots $|x_n - x_{n+1}| \approx 1/(\pi n^2) = \pi x_n^2$, $x \to 0$. As a test of the root-finding method, we seek roots starting from points increasingly close to the origin. A basic requirement of the method is that it should work until the distance between successive roots is comparable to machine precision, at which point roots become indistinguishable.

The test involved seeking ten roots moving toward the origin from a starting point $x_0 = 10^{-m}$, $m = 0, 1, \ldots, 7$. Roots found were checked to ensure that they corresponded to successive values of n in $x_n = 1/(n\pi)$. To allow for the increasing frequency of oscillation as $x_0 \to 0$, the tolerance for setting the step size h was set to $\varepsilon = x_0^2/10$, while the small increment required to shift the starting point away from a just-found root was set to $\delta = x_0^2/1000$. The measure of computational effort, shown in Fig. 3, is the total number of function evaluations in finding the ten roots, as a function of m and of the order of the Taylor polynomial. As expected, increasing the order of the method reduces the computational effort considerably—by more than two orders of magnitude for the cases considered here—and starting nearer the origin leads to an increase in effort as the roots become more closely spaced and the interval of validity of the Taylor polynomial becomes smaller so that more function evaluations are needed.

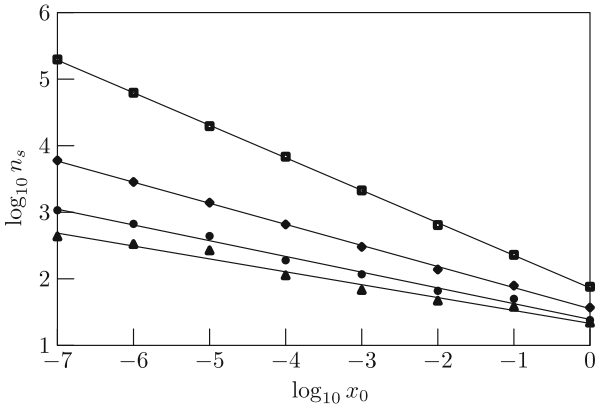

Fig. 3 Number of function evaluations n_s to find roots of $\sin(1/x)$ against starting point $x_0 = 10^{-m}$, *squares*: fourth order; *diamonds*: sixth order: *circles*: eighth order; *triangles*: tenth order; *solid lines*: linear fit $n_s = m^b$

In this test case, the tolerance is varied over many orders of magnitude $10^{-15} \leq \varepsilon \leq 10^{-3}$ in order to adapt to the increasing frequency of oscillation of the function, with $4 \leq N \leq 10$. In all cases, the method continued to give accurate answers with no difficulties arising in selecting a step size h.

3.3 Orthogonal SR–functions and para-orthogonal polynomials on the unit circle

Our final set of examples is made up of functions given in terms of a recursion relation. These functions are not polynomials so that many classical root-finding methods are ruled out, nor do we have enough information for use in an o.d.e. method [5, 8].

The functions in question have some orthogonality properties defined using a positive measure $\mathrm{d}\phi(x)$ on the interval $(-1, 1)$. A sequence of such functions $\{W_n(x)\}_{n=0}^{\infty}$ has been studied previously [3] using the orthogonality properties

$$\begin{aligned}
\int_{-1}^{1} W_{2n}(x)\, W_{2m}(x)\, \sqrt{1-x^2}\, \mathrm{d}\phi(x) &= \rho_{2m}\, \delta_{n,m}, \\
\int_{-1}^{1} W_{2n+1}(x)\, W_{2m+1}(x)\, \sqrt{1-x^2}\, \mathrm{d}\phi(x) &= \rho_{2m+1}\, \delta_{n,m}, \\
\int_{-1}^{1} W_{2n+1}(x)\, W_{2m}(x)\, \mathrm{d}\phi(x) &= 0,
\end{aligned} \tag{6}$$

for $n, m = 0, 1, 2, \ldots$, with $\rho_{2m}\rho_{2m+1} \neq 0$, $\delta_{n,m} = 0$,if $n \neq m$ and $\delta_{n,m} = 1$,if $n = m$.

It has been proved that the sequence of orthogonal functions $\{W_n(x)\}_{n=0}^{\infty}$ satisfies the following three–term recurrence relation

$$W_{m+1}(x) = \left(x - c_{m+1}\sqrt{1-x^2}\right) W_m(x) - d_{m+1} W_{m-1}(x), \quad m \geq 1, \tag{7}$$

with $W_0(x) = 1$, $W_1(x) = x - c_1\sqrt{1-x^2}$, where

$$c_{m+1} = \frac{1}{\rho_m}\int_{-1}^{1} x\, W_m^2(x)\, \mathrm{d}\phi(x), \quad m \geq 0 \qquad \text{and}$$

$$d_{m+1} = \frac{1}{\rho_{m-1}}\int_{-1}^{1} \left(x - c_{m+1}\sqrt{1-x^2}\right) W_{m-1}(x)\, W_m(x)\sqrt{1-x^2}\, \mathrm{d}\phi(x), \quad m \geq 1.$$

We refer to the functions $W_n(x)$ which satisfy (7) as SR-functions (*square-root*), to emphasize that they are not polynomial, though they do reduce to polynomials in the special case $c_{m+1} = 0$.

From the recurrence relation (7), one can see that an SR-function can be written as

$$W_n(x) = A_n(x) + \sqrt{1-x^2}B_{n-1}(x), \quad n \geq 0,$$

where $A_n(x) \in \mathbb{P}_n$ and $B_{n-1}(x) \in \mathbb{P}_{n-1}$ satisfy $A_n(-x) = (-1)^n A_n(x)$ and $B_{n-1}(-x) = (-1)^{n-1}B_{n-1}(x)$,and $\mathbb{P}_n$ denotes the space of the polynomials of degree at most n.

Our interest in these functions comes from the fact that in the interval $(-1, 1)$ an SR-function, $W_n(x)$, which satisfies the orthogonality properties (6) has exactly n simple zeros [3].

Furthermore, the orthogonal SR-functions have an interesting connection with orthogonal polynomials on the unit circle. Let $\mathrm{d}\mu(z)$ be a positive measure on the unit circle $\mathbb{T} = \{z = e^{\mathrm{i}\theta} : 0 \leq \theta \leq 2\pi\}$. Then, we call the sequence of polynomials $\{S_n(z)\}_{n=0}^{\infty}$ such that

$$\int_{\mathbb{T}} \bar{z}^k S_n(z) d\mu(z) = \int_0^{2\pi} e^{-\mathrm{i}k\theta} S_n(e^{\mathrm{i}\theta}) d\mu(e^{i\theta}) = 0, \quad 0 \leq k \leq n-1, \quad n \geq 1$$

a sequence of orthogonal polynomials on the unit circle (OPUC).

It is well known that [12] all zeros of these polynomials lie inside the unit disk and that they satisfy the relations

$$S_n(z) = zS_{n-1}(z) - \overline{\alpha}_{n-1}\, S_{n-1}^*(z), \quad n \geq 1,$$

where $\alpha_{n-1} = -\overline{S_n(0)}$ and the reciprocal polynomial of $S_n(z)$ is $S_n^*(z) = z^n\overline{S_n(1/\bar{z})}$.

Para-orthogonal polynomials on the unit circle (POPUC) associated with the sequence of OPUC $\{S_n(z)\}_{n=0}^{\infty}$ can be given by $zS_{n-1}(z) - \tau_n\, S_{n-1}^*(z)$, $n \geq 1$, where $\{\tau_n\}_{n=1}^{\infty}$ is any sequence of complex numbers such that $|\tau_n| = 1$. Unlike the OPUC, the POPUC has all its n zeros simple and lying on the unit circle $\mathbb{T}$.

If the positive measure $\mathrm{d}\phi(x)$ on $(-1, 1)$ is such that $\int_{-1}^{1}(1-x^2)^{-1/2}\mathrm{d}\phi(x)$ exists, then we can define $\mathrm{d}\mu(z)$, a positive measure on the unit circle, by

$$\mathrm{d}\mu(z) = -\mathrm{d}\phi(x(z))/\sin(\theta/2), \tag{8}$$

where $x(z) = (z^{1/2} + z^{-1/2})/2 = \cos(\theta/2)$, with $0 \leq \theta \leq 2\pi$ and $z = e^{\mathrm{i}\theta}$. From the three-term recurrence relation (7), it is easy to see that polynomials defined by

$$R_n(z) = 2^n z^{n/2} W_n(x(z)), \quad n \geq 0, \tag{9}$$

satisfy the three-term recurrence relation

$$R_{n+1}(z) = \left[(1 + \mathrm{i}c_{n+1})z + (1 - \mathrm{i}c_{n+1})\right] R_n(z) - 4\, d_{n+1} z R_{n-1}(z), \tag{10}$$

with $R_0(z) = 1$ and $R_1(z) = (1 + ic_1)z + (1 - ic_1)$. In [7], it was shown that functions $(z-1)R_n(z)$ are para-orthogonal on the unit circle, with $\tau_n = S_n(1)/S_n^*(1)$, associated with the measure $\mathrm{d}\mu(z)$ given in (8).

We must emphasize that, from relation (9), the zeros of the POPUC $(z-1)R_n(z)$ which lie on the unit circle and are used in quadrature rules on the unit circle [11] can easily be computed from $z_j = e^{\mathrm{i}\theta_j}, \theta_j = 2\arccos(x_j)$, where x_j, $j = 1, 2, \cdots, n$,are the zeros of the orthogonal SR–function $W_n(x)$.

We now use our proposed algorithm to seek roots of $W_n(x)$ for different measures $\mathrm{d}\phi(x)$. For our first example, we consider the positive measure $\mathrm{d}\phi(x) = e^{-2\eta\cos^{-1}(x)}(1-x^2)^{\lambda-1}\mathrm{d}x$, with $\eta, \lambda \in \mathbb{R}$ and $\lambda > 1/2$. In this case [3],

$$c_m = \frac{\eta}{m+\lambda-1} \quad \text{and} \quad d_{m+1} = \frac{m(m+2\lambda)-1}{4(m+\lambda-1)(m+\lambda)}, \quad m \geq 1.$$

Note that when $\eta = 0$, the orthogonal SR-functions $W_n(x)$ reduce to the monic ultraspherical polynomials $C_n^{(\lambda-1/2)}(x)$.

Varying λ and η gives quite different behaviour to the family of functions and can lead to quite testing conditions for a root-finding algorithm. Figure 4 shows the roots of $W_{24}(x)$ with $\lambda = 0.25$, $\eta = 0.9$. Roots were found working downwards from $x_0 = 1 - 10^{-7}$, with $\delta = 10^{-7}$. Other than the square-root behaviour at the end points, this is a reasonably well-behaved function without a large variation in amplitude. The roots are well-separated as is clear from Fig. 4.

Figure 5 is another matter: the function is of quite large amplitude for $x \lessapprox -1/2$ and is then exponentially small, with oscillations of increasing frequency as $x \to 1$, where the function has a square root singularity. Nonetheless, our method finds the roots in $-1 \leq x \leq 1$ accurately, as can be seen from the lower plot of Fig. 5 which gives the detail of the function and roots for $0.9 \leq x \leq 1$. The roots are correctly identified despite the small amplitude of the function (note the scaling on the vertical axis) and the high frequency oscillations.

Finally, Fig. 6 shows the computational effort required to find the roots in Fig. 5, as a function of the order N of the method. The reduction in effort, shown as the

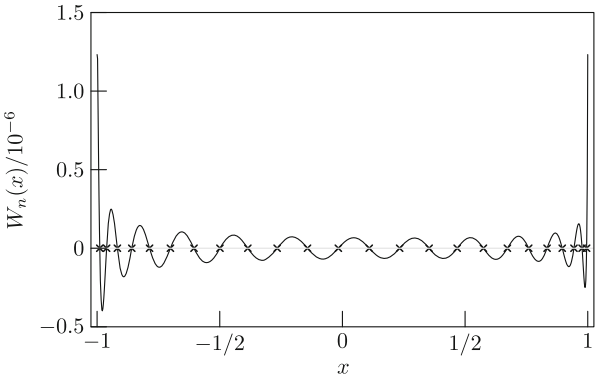

Fig. 4 Roots of SR-function $W_{24}(x)$, $\lambda = 0.25$, $\eta = 0.9$

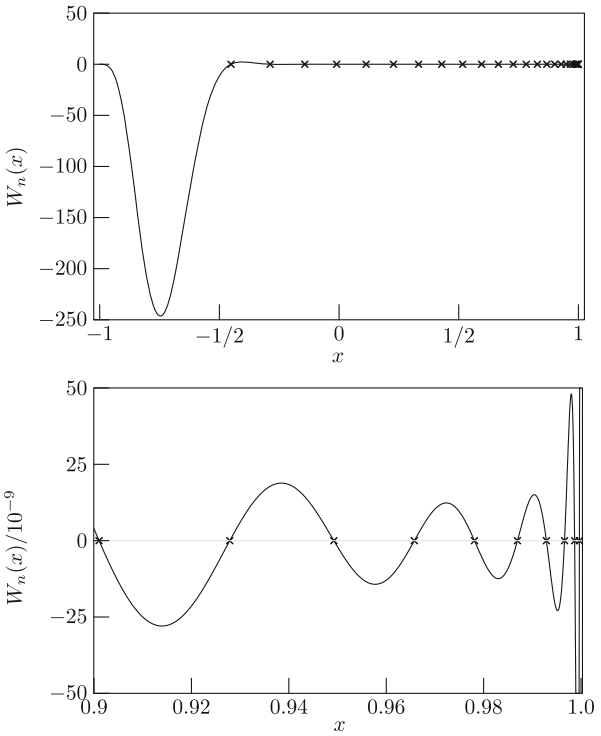

Fig. 5 Roots of $W_{25}(x)$, $\lambda = 13$, $\eta = 3$, *upper plot*: full set of roots in $(-1, 1)$; *lower plot*: zoom to $0.9 \leq x \leq 1$ (note scaling of vertical axis)

number of function evaluations required, is quite marked, falling by more than two orders of magnitude as N increases from 4 to 9, though the improvement is slower as N increases beyond this point.

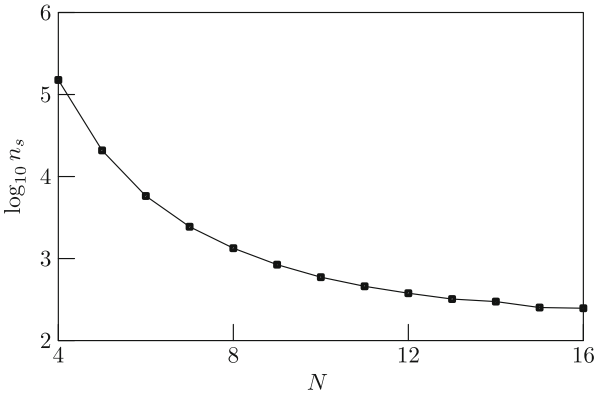

Fig. 6 Number of function evaluations to find roots in Fig. 5 against order of method

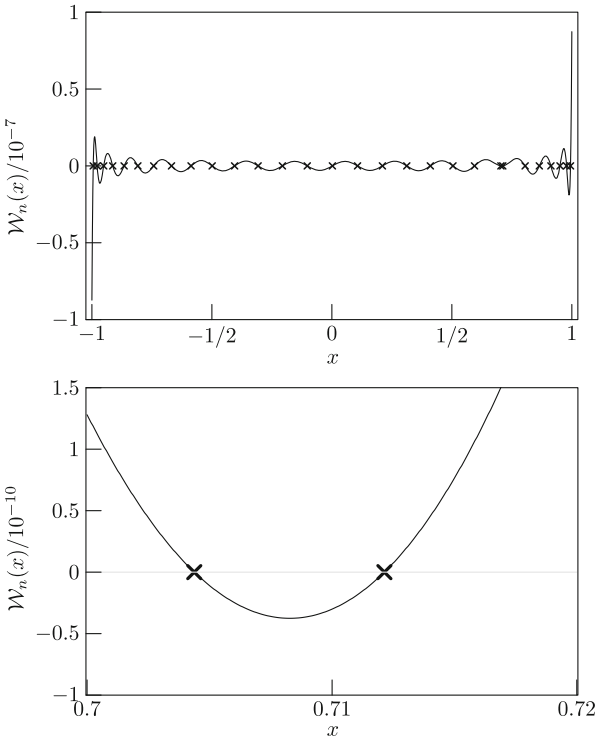

Fig. 7 Roots of $W_{29}(x)$, $\kappa = 0.8$: *upper plot*, all roots; *lower plot*, zoom to near-double root

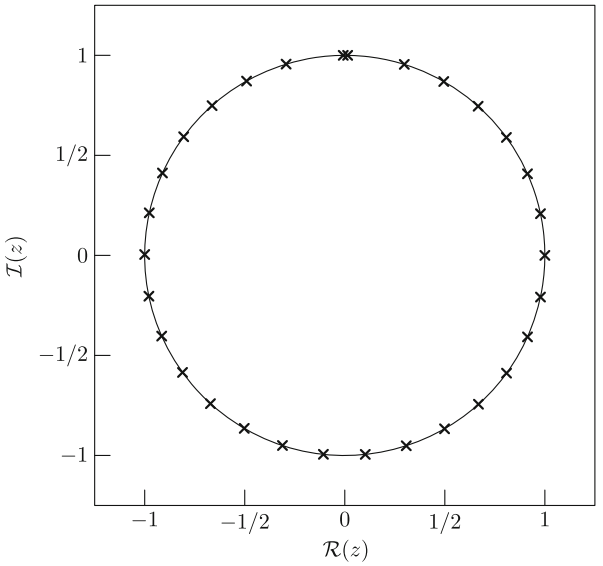

Fig. 8 Roots of $(z-1)R_{29}(z)$ on the unit circle for $\kappa = 0.8$

Our second example of an application considers para-orthogonal polynomials on the unit circle [4] associated with a Lebesgue measure with a mass κ at $z = \mathrm{i}$, which can be given as

$$\int_{\mathbb{T}} f(z)\mathrm{d}\mu^{(\kappa)}(z) = (1-\kappa)\int_{\mathbb{T}} f(z)\frac{1}{2\pi \mathrm{i}z}dz + \kappa f(\mathrm{i}) \tag{11}$$

where $0 \leq \kappa < 1$ and $\mathbb{T} = \{z = e^{i\theta} : 0 \leq \theta \leq 2\pi\}$ is the unit circle.

In [4, Theorem 5], considering the POPUC $(z-1)R_n(z)$ associated with the measure $\mathrm{d}\mu^{(\kappa)}(z)$, it was shown that the polynomials $R_n(z)$, satisfy the recurrence relation (10), with the coefficients $\{c_n\}_{n=1}^{\infty}$ and $\{d_{n+1}\}_{n=1}^{\infty}$ given, explicitly, by $d_{n+1} = (1-M_n)\,M_{n+1},\ n \geq 1$, and

$$\begin{aligned}
c_{4m+1} &= \frac{\kappa}{4m\kappa+1}, & M_{4m+1} &= \frac{1}{2}\frac{(4m\kappa+1)^2+\kappa^2}{(4m\kappa+1)^2},\\
c_{4m+2} &= \frac{-2\kappa^2}{[(4m+1)\kappa+1]^2}, & M_{4m+2} &= \frac{1}{2}\frac{(4m\kappa+1)\,[[(4m+2)\kappa+1]^2+\kappa^2]}{[(4m+1)\kappa+1]^3},\\
c_{4m+3} &= \frac{-\kappa}{(4m+2)\kappa+1}, & M_{4m+3} &= \frac{1}{2}\frac{[(4m+2)\kappa+1]^2-\kappa^2}{[(4m+2)\kappa+1]^2},\\
c_{4m+4} &= 0, & M_{4m+4} &= \frac{1}{2}\frac{(4m+2)\kappa+1}{(4m+3)\kappa+1},
\end{aligned} \tag{12}$$

for $m \geq 0$.

Then, using the relation (8) and definition (11), we see that the functions $W_n(x) = 2^{-n}z^{-n/2}R_n(z)$ satisfy the orthogonality conditions (6), where $\mathrm{d}\phi(x)$ is such that

$$\int_{-1}^{1} f(x)\mathrm{d}\phi(x) = (1-\kappa)\int_{-1}^{1} f(x)\mathrm{d}x + \kappa f(\sqrt{2}/2). \tag{13}$$

Furthermore, the orthogonal SR–functions $W_n(x)$ satisfy the recurrence relation (7) with the same coefficients $\{c_n\}_{n=1}^{\infty}$ and $\{d_{n+1}\}_{n=1}^{\infty}$ given in (12).

Figure 7 shows results of root–finding for $n = 29$, $\kappa = 0.8$. Roots were found using eighth-order Taylor polynomials, starting from $x_0 = -0.9999$, with $\delta = 10^{-7}$ and $\varepsilon = 10^{-12}$. From the upper plot, it is clear that the roots have been found successfully. The lower plot shows that, importantly, the algorithm has correctly detected and separated two roots which are close but distinct, near $x = \sqrt{2}/2$. This is an essential feature of any root-finding method, especially one used for orthogonal or para-orthogonal functions which form the basis of a quadrature rule.

Finally, to complete the calculation, Fig. 8 shows the roots of the POPUC $(z-1)R_n(z)$ for $n = 29$, associated with the measure $\mathrm{d}\mu^{(\kappa)}(z)$ given by (11) with $\kappa = 0.8$, computed from the real roots of $W_{29}(x)$, with the addition of the root $z = 1$.

4 Conclusions

We have presented a method which is well-suited to the problem of finding simple real roots of non-polynomial functions when a differential equation is not available for use in alternative algorithms. We have demonstrated that the method works on

some realistic problems, including some deliberately chosen to pose a challenge to root-finding methods. The technique is accurate and efficient, and can be implemented using standard coding tools. We note finally that the approach is easily modified to extract roots of first or other derivatives, such as function extrema, which are important in many applications, and that it is easily implemented on parallel systems.

Acknowledgments Work described in this paper was carried out while the second author was visiting UNESP under FAPESP contract 2014/17357-1, and while the first author was visiting the University of Bath under FAPESP contract 2014/22571-2.

References

1. Bornemann, F.: Accuracy and stability of computing high-order derivatives of analytic functions by Cauchy integrals. Found. Comput. Math. **11**(1), 1–63 (2011). doi:10.1007/s10208-010-9075-z
2. Boyd, J.P.: Finding the zeros of a univariate equation: Proxy rootfinders, Chebyshev interpolation and the companion matrix. SIAM Rev. **55**(2), 375–396 (2013). doi:10.1137/110838297
3. Bracciali, C.F., McCabe, J.H., Pérez, T.E., Sri Ranga, A.: A class of orthogonal functions given by a three term recurrence formula. Math. Comput. **85**(300), 1837–1859 (2016). doi:10.1090/mcom3041
4. Bracciali, C.F., Silva, J.S., Sri Ranga, A.: Explicit formulas for OPUC and POPUC associated with measures which are simple modifications of the Lebesgue measure. Appl. Math. Comput. **271**, 820–831 (2015). doi:10.1016/j.amc.2015.09.067
5. Bremer, J.: On the numerical calculation of the roots of special functions satisfying second order ordinary differential equations. SIAM J. Sci. Comput. **39**(1), A55–A82 (2017). doi:10.1137/16M1057139
6. Bremer, J., Rokhlin, V.: Improved estimates for nonoscillatory phase functions. Discret. Contin. Dyn. Syst. **36**(8), 4101–4131 (2016). doi:10.3934/dcds.2016.36.4101
7. Costa, M.S., Felix, H.M., Sri Ranga, A.: Orthogonal polynomials on the unit circle and chain sequences. J. Approx. Theory **173** (2013). doi:10.1016/j.jat.2013.04.009
8. Glaser, A., Liu, X., Rokhlin, V.: A fast algorithm for the calculation of the roots of special functions. SIAM J. Sci. Comput. **29**(4), 1420–1438 (2007). doi:10.1137/06067016X
9. Gradshteyn, I., Ryzhik, I.M.: Table of integrals, series, and products, 5th edn. Academic, London (1980)
10. Hook, D.G., McAree, P.R.: Using Sturm sequences to bracket real roots of polynomial equations. In: Glassner, A.S. (ed.) Graphics Gems. Academeic Press, London (1990)
11. Jones, W.B., Njåstad, O., Thron, W.J.: Moment theory, orthogonal polynomials, quadrature, and continued fractions associated with the unit circle. Bull. Lond. Math. Soc. **21**(2), 113–152 (1989). doi:10.1112/blms/21.2.113
12. Simon, B.: Orthogonal polynomials on the unit circle: Part 1. Classical theory, vol. 54. American Mathematical Society Colloqium Publications (2005)
13. Sturm, C.: Analyse d'un mémoire sur la résolution des équations numériques. In: Pont, J.C. (ed.) Collected Works of Charles François Sturm, pp. 345–390, Birkhäuser Basel, Basel. doi:10.1007/978-3-7643-7990-2_24. Originally published 1835 (2009)